Costing Models for Information Technology (IT) Services

Dr. Ashish Varma*

Abstract

The cost estimation process includes a number of iterative steps. The reason for the iteration over the different steps is that cost estimation is part of the larger planning and design process, in which the system is designed to fit performance, cost, and schedule constraints along with reconciliation and review of the different estimates.

In the highly competitive Information Technology (IT) sector, firms need to be cost efficient in order to remain competitive.

1. Introduction

In the highly competitive Information Technology (IT) sector, firms need to be cost efficient in order to remain competitive. The paper aims to describe costing process generally followed by IT Services Company and will focus on estimation effort of software.

The prescribed method applies to the estimation of the costs associated with the software development portion of a project from software requirements analysis, design, coding, software integration and test (I&T), through completion of system test (Neumann et al 2004). Other Activities like software management, configuration management (CM), and software quality assurance, as well as other costs, such as hardware (HW) procurement costs and travel costs, that also part of IT project is kept out of scope (James et al 2002).

For the purpose of the paper, the author studied the costing practices of 4 leading Indian IT companies and the existing literature on the subject in journals and books. The learning has presented a model which is generally followed by all.

The following two are the main Pricing Model followed in IT Services Company

Assistant Professor, IMT, Ghaziabad.

1. **Fixed Price Fixed Time**
   a. customer pays a pre-negotiated fixed price for the complete project
   b. When project specifications, scopes, deliverables and acceptance criteria are clearly defined
   c. For any change "change request procedure is followed" and impact is valuated

2. **Time and Material Pricing**
   a. Customer pays as per resources used for development
   b. When scope, specification and implementation plans of a software development project are not easy to define at the outset
   c. Work hand-in-hand with client for execution of end-to-end project

2. Cost Estimation: Approach and Methods

The purpose of software cost estimation is to:

1. Ascertain the resources needed to produce, verify, and validate the software product, and manage these activities.
2. Measure the uncertainty and risk inherent in this estimate.

For software development, the dominant cost is the cost of labor. Therefore, it is very important to estimate the software development effort as accurately as possible. A basic cost equation for the costs covered in most handbooks can be defined as:

Total_SW_Project = SW_Development_Labor + Other_Labor$ + Nonlabor

The software development Labour includes:

- **Software Systems Engineering** - performed by the software architect, software system engineer, and subsystem engineer for functional design, software requirements, and interface specification. Labor for data systems engineering, which is often forgotten, should also be considered. This includes science product definition and data management.
- **Software Engineering** - performed by the cognizant engineer and developers to unit design, develop code, unit test, and integrate software components
- **Software Test Engineering** – covers test engineering activities from writing test plans and procedures to performing any level of test above unit testing

3. Estimation Methods

All estimates are made based upon some form of analogy: Historical Analogy, Expert Judgment, Models, and "Rules-of-Thumb." The role these methods play in generating an estimate depends upon where one is in the overall life-cycle.

Typically, estimates are made using a combination of these four methods (Neumann et al 2004). Model-based estimates along with high-level analogies are the principal source of estimates in early conceptual stages. As a project matures and the requirements and design are better understood, analogy estimates based upon more detailed functional decompositions become the primary method of estimation, with model-based estimates used as a means of estimate validation or as a "sanity-check."

1. Historical analogy estimation methods are based upon using the software size, effort, or cost of a comparable project from the past. When the term "analogy" is used in this document, it will mean that the comparison is made using measures or data that has been recorded from completed software projects.
2. Expert judgment estimates are made by the estimator based upon what he or she remembers it took previous similar projects to complete or how big they were. This is typically a subjective estimate based upon what the estimator remembers from previous projects and gets modified mentally as deemed appropriate.
3. Model-based estimates are estimates made using mathematical relationships or parametric cost models. Parametric cost models are empirical relationships derived by using statistical techniques applied to data from previous projects.
4. "Rules-of-thumb" come in a variety of forms and can be a way of expressing estimates as a simple mathematical relationship (e.g. Effort = Lines_of_Code / 10) or as percentage allocations of effort over activities or phases based upon historical data (e.g. I&T is 22% of Total Effort).

4. Software Estimation Steps

The cost estimation process includes a number of iterative steps summarized in Table 1. The reason for the iteration over the different steps is that cost estimation is part of the larger planning and design process, in which the system is designed to fit performance, cost, and schedule constraints along with reconciliation and review of the different estimates. Although, in practice, the steps are often performed in a different order and are highly iterative, these steps will be discussed in the sequence that they are numbered for ease of exposition and because this is one of the ideal sequences (Brewer 1998)

Software project plans include estimates of cost, product size, resources, staffing levels, schedules, and key milestones. The software estimation process discussed in the following subsections describes the steps for developing software estimates. Establishing this process early in the life-cycle will result in greater accuracy and credibility of estimates and a clearer understanding of the factors that influence software development costs. This process also provides methods for project personnel to identify and monitor cost and schedule risk factors.

Table 1 gives a brief description of the software estimation steps (Boehm et al 2002). Projects define which personnel are responsible for the activities in the steps. Table 1 presents the roles of personnel who typically perform the activities in each step. The participants should have experience similar to the software under development.

Table 1

Action	Description	Responsibility	Output Summary
Step 1: Gather and Analyze Software Functional & Programmatic Requirements	Analyze and refine software requirements, software architecture, and programmatic constraints.	Software manager, system engineers, and cognizant engineers.	• Identified constraints • Methods used to refine requirements • Resulting requirements
Step 2: Define the Work Elements and Procurements	Define software work elements and procurements for specific project.	Software manager, system engineers, and congnizant engineers.	• Project-Specific product-based software WBS • Procurements • Risk List
Step 3: Estimate Software Size	Estimate size of software in logical Source Lines of Code (SLOC).	Software manager, cognizant engineers.	• Methods used for size size estimation • Lower level and total software size estimates in logical SLOC.
Step 4: Estimate Software Effort	Convert software size estimate in SLOC to software development effort. Use software development effort to derive effort for all work elements.	Software manager, cognizant engineers, and software estimators.	• Methods used to estimate effort for all work elements • Lower level and Total Software • Development Effort in work-months (WM)
Step 5: Schedule the effort	Determine length of time needed to complete the software effort. Establish time periods of work elements of the software project WBS and milestones.	Software manager, cognizant engineers, and software estimators.	• Schedule for all work elements of project's software WBS • Milestones and review dates • Revised estimates and assumptions made
Step 6: Calculate the Cost	Estimate the total cost of the software project.	Software manager, cognizant engineers, and software estimators.	• Methods used to estimate the cost • Cost of procurements
Step 7: Determine the Impact of Risks	Identify software project risks, estimate their impact, and revise estimates.	Software manager, cognizant engineers, and software estimators.	• Detailed Risk List • Methods used in risk estimation • Revised size, effort, and cost estimates
Step 8: Validate and Reconcile the Estimate Via Models and Analogy	Develop alternate effort, schedule, and cost estimates to validate original estimates and to improve accuracy.	Software manager, cognizant engineers, and software estimators.	• Methods used to validate estimates • Validated and revised size, effort, schedule, and cost estimates.
Step 9: Reconcile Estimates, Budget, and Schedule	Review above size, effort, schedule, and cost estimates and compare with project budget and schedule. Resolve inconsistencies.	Software manager, software engineers, software estimators, and sponsors.	• Revised size, effort, schedule, risk and cost estimates • Methods used to revise estimates
Step 10: Review and Approve the Estimates	Review and approve software size effort, schedule, and cost estimates.	The above personnel, software engineer with experience on similar project, line and project management.	• Problems found with reconciled estimates • Reviewed, revised, and approved size, effort, schedule, and cost estimates • Work agreement(s), if necessary
Step 11: Track, Report, and Maintain the Estimates	Compare estimates with actual data. Track estimate accuracy. Report and maintain size, effort, schedule, and cost estimates at each major milestone.	Software manager, software engineers and software estimators	• Evaluation of comparisons of actual and estimated data

Table 2: Converting Size Estimates

Language	To Derive Logical SLOC
Assembly and Fortran	Assume Physical SLOC = Logical SLOC
Third-Generation Languages[3] (C, Cobol, Pascal, Ada 83)	Reduce Physical SLOC by 25%
Fourth-Generation Languages[3] (e.g., SQL, Perl, Oracle)	Reduce Physical SLOC by 40%
Object-oriented Languages[3] (e.g., Ada 95, C++, Java, Python)	Reduce Physical SLOC by 30%

Table 3: Software Development Productivity for Industry Average Projects

Characteristic	Software Development Productivity (SLOC/WM)
Classical rates	130 – 195
Evolutionary approaches	244 – 325
New embedded flight software	17 - 105

Adjust the effort estimates of each software function for software heritage by multiplying the Software Development Effort by the effort multiplier according to following table:

Table 4: Effort Adjustment Multipliers for Software Heritage

Software Heritage Category	Effort Multiplier
New design and new code	1.2
Similar design and new code (nominal case)	1.0
Similar design and some code reuse	0.8

Table 5: Software Cost Risk Drivers and Ratings

Risk Drivers	Software Cost Risk Driver Ratings	
	Nominal (Reduces Risk)	Extra High (Increases Risk)
Experience & Teaming	• Extensive software experience in the project office • Software staff included in early planning and design decisions • Integrated HW and SW teams	• Limited software experience in the project office • Software staff not included in early planning and design decisions • HW and SW teams are not integrated
Planning	• Appropriately detailed and reviewed Plan • All key parties provide input with time to get buy-in • Appropriate assignment of reserves • SW inheritance verified based on review and adequate support	• Lack of appropriate planning detail with insufficient review • Not all parties involved in plan development • Simplistic approach to reserve allocation • Optimistic non-verified assumptions especially with respect to software inheritance
Requirements & Design	• Solid system and SW architecture with clear rules for system partitioning • Integrated systems decisions based on both HW and SW criteria • SW Development process designed to allow for evolving requirements	• System and Software architecture not in place early with unclear descriptions of basis for HW & SW partitioning of functionality. • Systems decisions made without accounting for impact on software • Expect SW requirements to solidify late in the life-cycle
Staffing	• Expected turnover is low • Bring software staff on in timely fashion • Plan to keep software team in place through launch	• Expected turnover is high • Staff up software late in life-cycle • Plan to release software team before ATLO
Testing	• Multiple Test-beds identified as planned deliverables and scheduled for early completion. • Separate test team • Early development of test plan	• Insufficient Test-beds/simulators dedicated to SW & are not clearly identified as project deliverables • Plan to convert SW developers into test team late in life-cycle • Test documents not due till very late in the life- cycle

Tools	• CM and Test tools appropriate to project needs • Proven design tools	• No or limited capability CM and test analysis tools • Unproven design tools selected with limited time for analysis

Table 6: Estimate Cost Impact of Risk Drivers for High-Plus Ratings

Risk Drivers	Estimated Cost Impact		
	High	Very High	Extra High
Experience & Teaming	1.02	1.05	1.08
Planning	1.10	1.17	1.25
Requirements & Design	1.05	1.13	1.20
Staffing	1.02	1.05	1.13
Testing	1.05	1.08	1.15
Tools	1.02	1.03	1.10
Maximum Expected Cost Impact	1.30	1.60	2.32

- External Input – None
- External output – the report count:1
- Logical Internal File – None
- External Interface Files – payroll, staff, course tables count:3
- External Inquiry – None

Total Functional Points Count=(1x7)+(3x7) = 28
Calculate of SLOC
Adjustment Factor: ASP 32-56, C++ 20-59, Java 9-55, VB.NET 28
Estimated SLOC for Cobol = 28 x 91 = 2548
Productivity rate = 50 SLOC per day
Estimated Effort = 2548 / 50 = 51 days

Project complexity	Formula	Description
Simple	PM = 2.4 (KDSI)^1.05 TDEV=2.5 (PM)^0.38	**Well-understood applications developed by small teams.**
Moderate	PM = 3.0 (KDSI)^1.12 TDEV=2.5 (PM)^0.35	**More complex projects where team members may have limited experience of related systems.**
Embedded	PM = 3.6 (KDSI)^1.20 TDEV=2.5 (PM)^0.32	**Complex projects where the software is part of a strongly coupled complex of hardware, software, regulations and operational procedures.**

- Simple project, 32KLOC
  - PM = 2.4 (32) $^{1.05}$ = 91 person months
  - TDEV = 2.5 (91) $^{0.38}$ = 14 months
  - N = 91 / 15 = 6.5 people

- Embedded project, 128KLOC
  - PM = 3.6 (128)$^{1.2}$ = 1216 person-months
  - TDEV = 2.5 (1216)$^{0.32}$ = 24 months
  - N = 1216 / 24 = 51

- Illustration of the above:
  - Use Case: Login functionality on a site
  - Stories:

- User Interface design = 2 days = 18 hours

- Database design = 3 days = 27 Hours

- Middle layer using Java / J2EE = 7 days = 63 Hours

- Testing = 5 days = 45 hours

Total Use Case effort = 18 + 27 + 63 + 45 = 153 hours

- Say, effort for "Use Case 2" = 190 hours
- Project effort = 190 + 153 = 343 Hours
- Buffer = 40% * 343 Hours = 137 Hours
- Total Project Cost = 343 + 137 = 480 hours
- Now, its estimated how many Senior software engineers, software engineers, Technology Manager, Project manager are required for this effort
- It resource has a different cost. For example Manager will have a higher cost than that of software engineer or a Senior Software engineer.
- Cost of all these team members is calculated by multiplying with number of hours of effort they are expected to spend.
- Total cost is arrived at by adding cost of all team members.
- Per Hour cost of each team member includes direct cost as well as overheads.

5. Conclusion

The paper helps describe the costing process generally followed by IT Services Company and focuses on estimation effort of software. The Information Technology (IT) services have intangibility which makes their costing difficulty (Neuman 2004). Cost management is possible when the steps are clearly defined and understood.

The processes need to be modelled accurately and reliably. The solution then is useful to the managers for measuring the work and aiming for improvements.

6. References

1. *Neumann, Bruce R; Gerlach, James H; Moldauer, Edwin; Finch, Michael; Olson, Christine (2004). "Cost management using ABC for IT activities and Services" Management Accounting Quarterly 6. 1 (Fall 2004): 29.*
2. *Gerlach, James; Neumann, Bruce; Moldauer, Edwin; Argo, Martha; Frisby, Daniel. (2002) "Determine the cost of IT services" Association for Computing Machinery.*
3. *Business Wire [New York] 10 Apr 2006 Service based costing models Communications of the ACM 45. 9 (Sep 2002): 61-67*
4. *http://www.sei.cmu.edu/- Software Engineering Institute (SEI), - DOD FFRDC at Carnegie Mellon University focusing on software*
5. *http://sunset.usc.edu/ - USC Center for Software Engineering homepage and site for COCOMO family of cost models*
6. *http://www.ispa-cost.org/ - International Society of Parametric Analysts*
7. *http://users.erols.com/scea/ - Society of Cost Estimating and Analysis*
8. *http://www.spr.com/index.htm - Capers Jones' Software Productivity Research*
9. *Boehm, et al. Software Cost Estimation with COCOMO II. Prentice Hall, Upper Saddle River, N.J., 2000*
10. *Brewer, Peter C "Developing a data center chargeback system using ABC". Cost Management 12. 3 (May/Jun 1998): 41-47*